GATE LearningFramework

Neural Network Support: Outlook

June 2018

NNs for LF

- Neural Networks (NNs) / Deep Neural Networks (DNNs) have become important for NLP
- Plan: add NNs/DNNs to LearningFramework
- Ideally, should be as easy to use as existing algorithms:
  - Define features to use
  - Select ML algorithm
  - Press a button and wait ...
  - Use the model
- Turns out: not quite that simple!
- Why? Major differences between "classic" ML and NN-ML

Classic ML

- Our pipeline creates lots of good features
- Features get converted to sparse vector
  - Numeric values get represented as numeric values
  - Nominal values get represented by as many dimensions as there are values (for tokens: tens of thousands), only one of those dimensions is non-zero (=sparse)
  - Value is frequency or tf-idf (valuefrom)
  - `Token!stringLC!L-2!have=1,Token!upos!L-0!VERB=1.0 ...`
- Good features, right algorithm choice are very important
- Switch algorithms on same training set to find best
- Hyperparms not so much, only few of form `-p val`

NN ML

- No different algorithms, instead:
- Different frameworks
- Different architectures / network graphs
- Different loss functions, optimizers
- Regularization, attention, CRF layer, GANs, ...
- Many hyperparms for all of these
- Need/allow huge corpora: not in memory! Maybe need GPU to process!
- Features less important, often just "the text"
- Dense representation
- Other kinds of features: char embeddings
- Need much more experimentation, tuning, exploration

NN ML Dense Representation

- Numeric values are represented as numeric values
- Nominal values are mapped to a value index, each index is mapped to an “embedding” (a dense vector of numbers)
- Low number of nominal values can also get mapped to ”one-hot” vectors
- Embeddings can get learned as part of the training or pre-trained Embeddings get used, or a combination

NN ML Architecture

- Feed-forward (FF), Convolutional (CNN), Recursive (RNN)
- Number of layers, non-linearities, drop-out
- CNN: features, size, stride, pooling
- RNN: LSTM, GRU

NN Libraries

- Many NN libraries exist: Tensorflow (Google) Keras (wraps Tensorflow, Theano, CNTK) Lasagne (wraps Theano) Theano (RIP!) Torch, Pytorch Caffe (Berkely VLC), Caffe2 (Facebook) Chainer DyNet (Carnegie Mellon) CNTK (Microsoft) MxNet (Amazon) DeepLearning4J Paddle (Baidu) DSSTNE, BigDL, Gensim, ...
- Almost all written in C/C++/Python
- Almost all try to support GPUs
- We have to divide the work between GATE and the Library somehow: Wrapper!

Choice

- Keras:
  - + Based on Tensorflow but much easier, high-level, CNTK?
  - + Better save/load
  - + Easy installation as Python package
  - + Professional IT experts use it
  - – Static graphs
- PyTorch:
  - + Fast C++ foundation
  - + ! Dynamic graphs (recent SOA!)
  - + Built-in CUDA support
  - + Easy installation
  - + All the cool kids use it

Status: alpha1

- + LF: Dense Feature Extraction, additional options for embeddings
- + LF: Out of memory dense corpus representation
- + LF: bridge between GATE and Python programs (currently: Linux/Mac only)
- + Python: library `gate-lf-python-data`: process dense corpus, convert, batch-iteration, validation split
- + Python: library `gate-lf-keras-json`: NN Keras
- + Python: library `gate-lf-pytorch-json`: NN Pytorch
- - Auto-model generation
- - Model zoo
- - Documentation

Demos

- Demo 1: Ionosphere Classification
- Demo 2: English UPOS using SEQ
- Demo 3: English UPOS using FF and window
- Demo 4: Sentence Classification using N-grams